Propedéutico Especialidad en Sistemas Embebidos

Departamento de Electrónica, Sistemas e Informática

Tarea 2. Task Scheduler

Alumna:

Martha Angélica Mercado Ramos

~~Aldo Jorge Campos Pérez~~

Profesor: Abraham Tezmol Otero

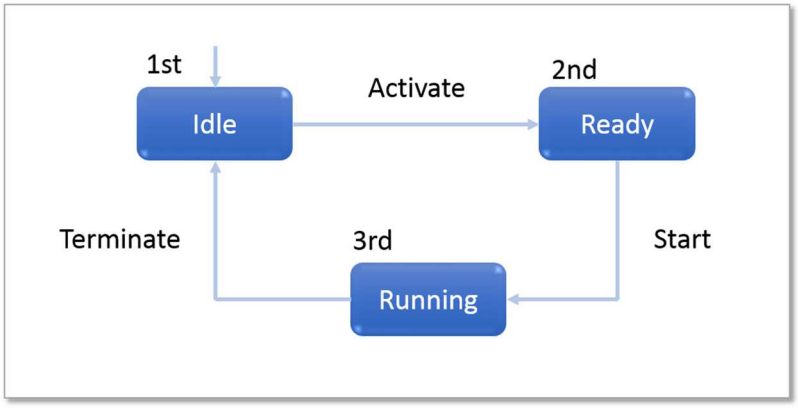
Fecha: 16 de Junio de 2018.

Introducción

Task

Las tareas pueden ser síncronas o asíncronas. Función o cualquier pieza de código que sea administrada por el Sistema Operativo.

Ciclo de vida de una tarea



Activate – Proceso síncrono o asíncrono, por él se cumplen las condiciones para que una tarea esté lista para ser ejecutada.

Start – Proceso por el cual el task scheduler realiza un barrido de las tareas que se encuentran en el estado Ready y despacha a la que tiene mayor prioridad para ser ejecutada.

Terminate – Proceso por el cual el task scheduler al terminar la tarea la pasa al estado Idle.

Schedulers

Es el centro de la funcionalidad del sistema operativo, ya que da la ilusión de que todas las tareas se están ejecutando simultáneamente. Esto se logra permitiendo que cada una tareas tenga una parte del tiempo del procesador. La manera en la que el tiempo es asignado entre las tareas es denominada “scheduling”. El scheduler determina cual es la siguiente tarea que debe ejecutarse después. La lógica del scheduler y el mecanismo que determina cuando debe ejecutarse es el ‘scheduling algorithm’.

Características de un Scheduler

- Proporciona un Sistema Operativo estable en los Sistemas Embebidos.
- Debe garantizar que en el peor de los casos el tiempo de respuesta para que el sistema operativo dé el control a un proceso que necesita atención, sea lo suficientemente corto para manejar los eventos.
- Prioriza los procesos para asegurar el tiempo de respuesta, siendo los más importantes aquellos que reciben atención por parte del procesador, si es que la necesitan.

Task Scheduling

Es la parte del Sistema Operativo que responde a las llamadas de programas e interrupciones, para llamar la atención del procesador y darle el control al mismo para procesarlas.

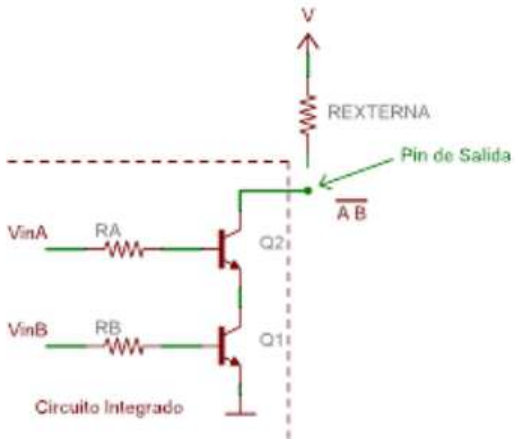
Scheduling Algorithm

Es el algoritmo que se sigue para decidir quién es el que tiene el siguiente turno en el CPU. El programa que sigue dicho algoritmo es llamado Scheduler.

Información requerida para lograr la activación de uno de los LEDs y realizar las mediciones

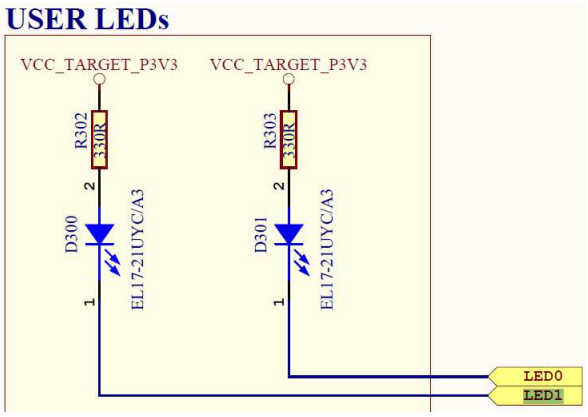
La configuración usada es llamada Open Collector o Open Drain, que son un tipo de compuertas lógicas cuya salida está abierta o sin resistencia en el colector del transistor de salida. Al utilizar ésta configuración se permite la posibilidad de utilizar el valor de resistencia apropiado según las necesidades.

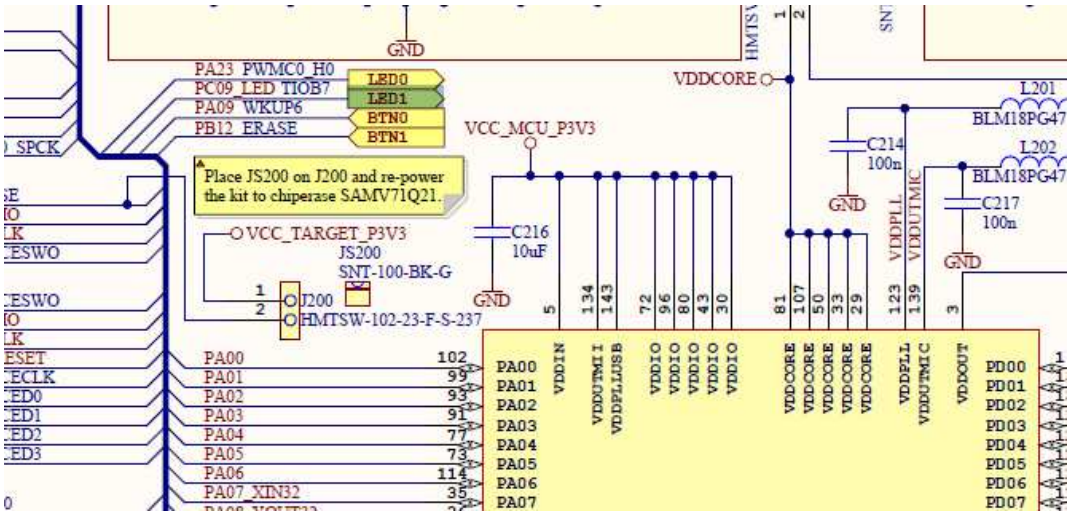
Este tipo de salidas son útiles para fijar los valores altos y bajos de tensión según las necesidades. Garantiza la corriente de salida necesaria.



LED0 / LED1

Las siguientes imágenes pertenecen al layout del board. En ellas es posible ver la configuración usada para los LEDs y los pines relacionados a ellos. En ésta ocasión el LED utilizado fue el LED1 – Pin PC09.





4.3.3. LEDs

There are two yellow LEDs available on the SAM V71 Xplained Ultra board that can be turned on and off. The LEDs can be activated by driving the connected I/O line to GND.

Table 4-22 LED Connection

SAM V71 pin	Function	Shared functionality
PA23	Yellow LED0	EDBG GPIO
PC09	Yellow LED1	LCD and Shield

Objetivos

Realizar los siguientes incisos tomando como base el proyecto “Task Scheduler”:

a) Definir en el código los puntos exactos en los que las diferentes tareas son activadas.
b) Definir en el código los puntos exactos en los que las diferentes tareas son ejecutadas.
c) Agregar la activación de un pin de salida (puede ser un LED) para medir el tiempo entre la activación de una tarea y la ejecución de la misma.
d) Repetir las mediciones para todas las tareas.

Desarrollo de la solución

Primero se incluyó el archivo led_ctrl.h dentro del app_scheduler.c, lo cual permite el uso de las funciones LED_Set() y LED_Clear(), enviando como parámetro el número de LED que va a ser usado para ser encendido o apagado.

startup_sam.c | led_ctrl.c | app_tasks.h | main.c | app_scheduler.c

```
10 /** Variable types and common definitions */
11 #include "system_samv71.h"
12
13 /** Scheduler function prototypes definitions */
14 #include "app_scheduler.h"
15 /** Tasks definition */
16 #include "app_tasks.h"
17 /** LED CONTROL - AnMer */
18 #include "led_ctrl.h"
```

Dentro del mismo app_scheduler.c, revisamos cada una de las partes del código con el objetivo de detectar aquella en la que se realiza el proceso de ACTIVACIÓN de la misma y colocamos el LED_Set(1).

Por ejemplo: En la siguiente imagen es posible apreciar que estamos identificando la parte de código en la cual la tarea de 10ms está siendo activada (dentro del contador que permite la ejecución por medio de la tarea de 2ms_b).

startup_sam.c | led_ctrl.c | app_tasks.h | main.c | app_scheduler.c

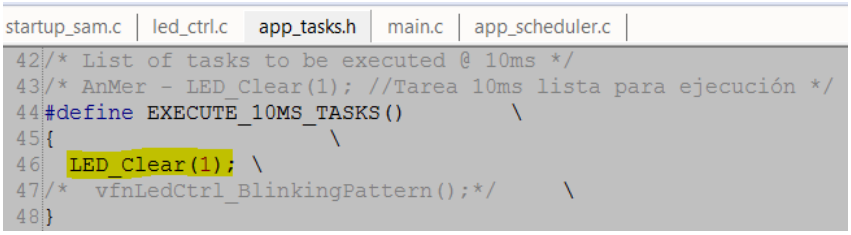
```
256       /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
257       /*  2ms execution thread - used to derive two execution threads:
258       /*  a) 2ms group B thread (high priority tasks)
259       /*  b) 10ms thread (medium priority tasks)
260       /*  As any other thread on this scheduling scheme,
261       /*  all tasks must be executed in <= 500us
262       /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
263       if ((gu8Scheduler_Counter & 0x03) == 0x00)
264       {
265         u8_10ms_Counter++;
266         /*-- Allow 10 ms periodic tasks to be executed --*/
267         if (u8_10ms_Counter >= 5)
268         {
269           gu8Scheduler_Thread_ID = TASKS_10_MS;
270           u8_10ms_Counter = 0;
271           LED_Set(1); /* AnMer - Tarea Activada */
272         }
273         /*-- Allow 2 ms group B periodic tasks to be executed --*/
```

Si fuera el caso de una tarea no dependiente de otra, como la de 2ms_b, entonces colocaríamos el LED_Set(1) inmediatamente después de que a la variable global gu8Scheduler_Thread_ID se le asigna el valor TASK_2_MS_B para que dicha tarea sea ejecutada.

led.c | led_ctrl.h | startup_sam.c | led_ctrl.c | app_tasks.h | main.c | app_scheduler.c *

```
273         /*-- Allow 2 ms group B periodic tasks to be executed --*/
274         else
275         {
276           gu8Scheduler_Thread_ID = TASKS_2_MS_B;
277           LED_Set(1); /* AnMer - Tarea Activada */
278         }
```

Por último dentro del app_task.h file, identificamos la tarea que va a ser ejecutada (por ejemplo: tarea de 10ms) y dentro de la misma colocamos el LED_Clear(1) para identificar el momento en el que la tarea está lista para ser ejecutada.



El procedimiento descrito anteriormente se repite para cada una de las tareas.

Pruebas

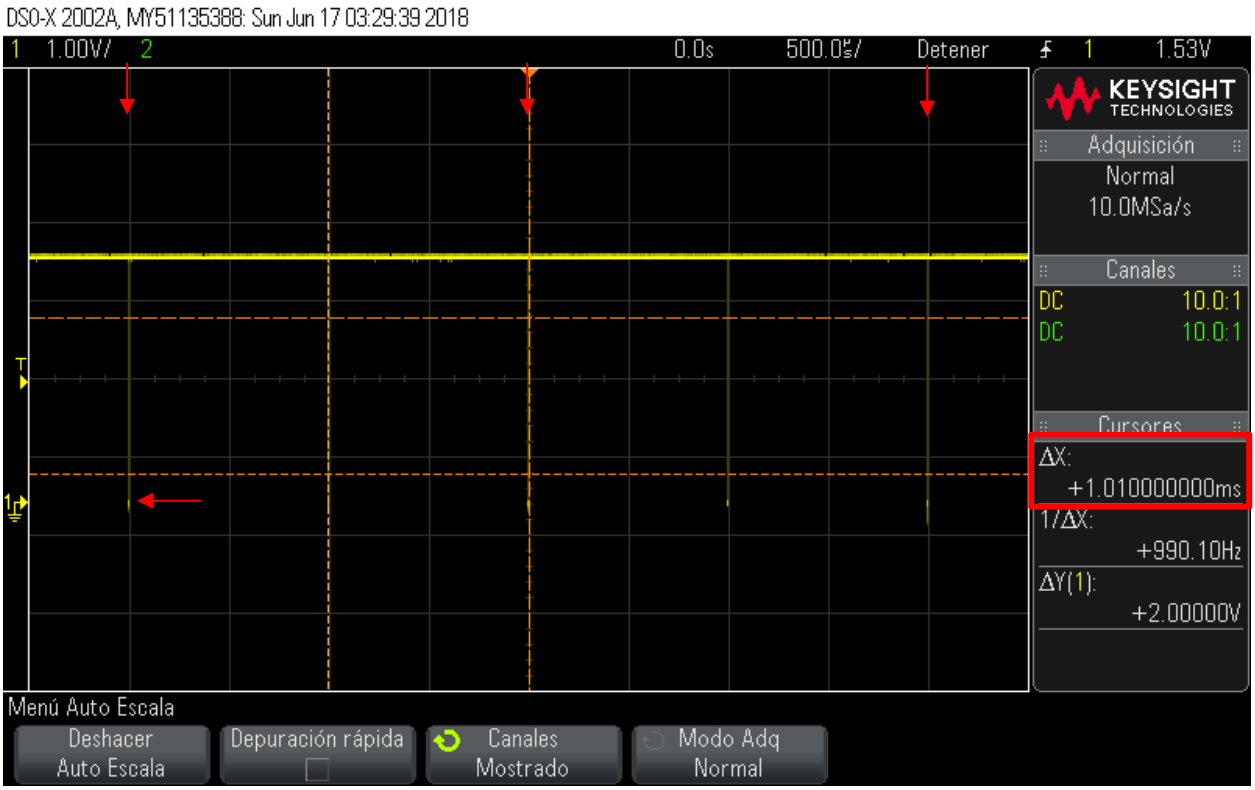
Una vez que identificamos los puntos exactos en que las distintas tareas son activadas y ejecutadas.

Recompilamos el proyecto.

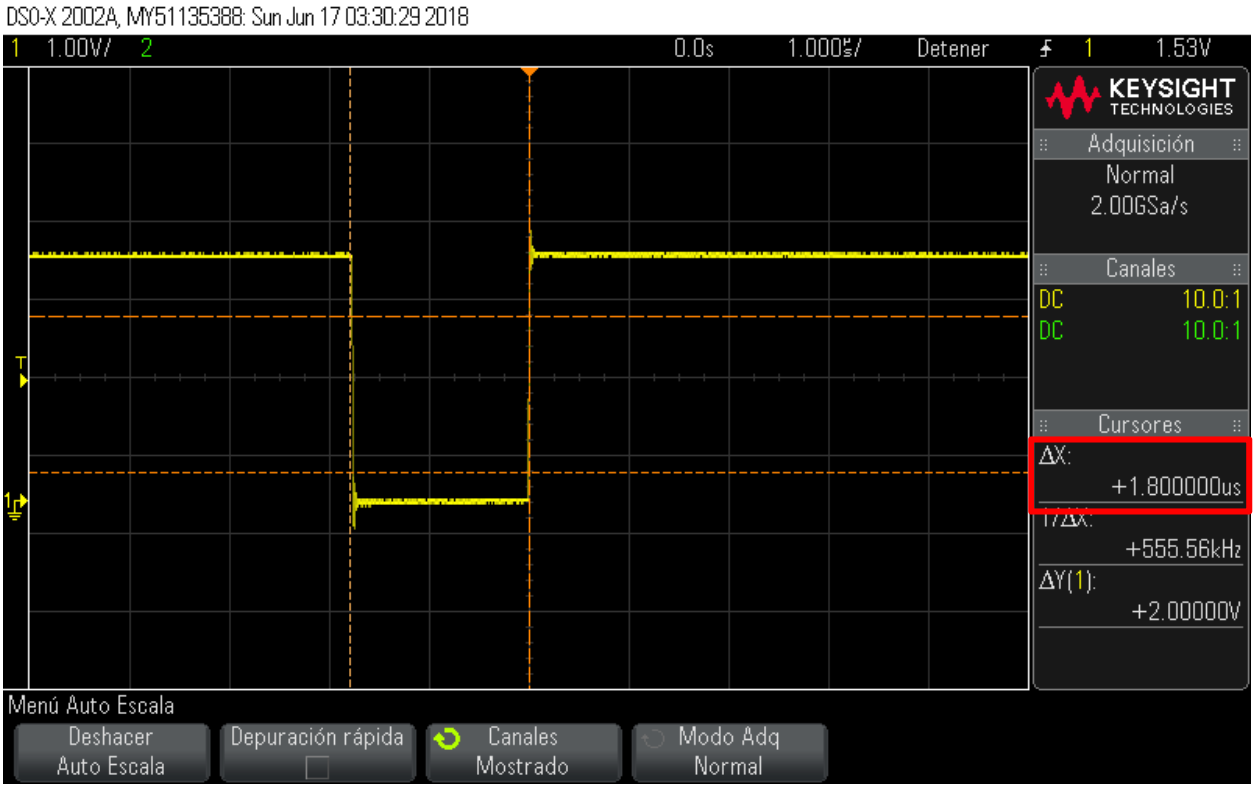
Cargamos a la tarjeta el programa y lo corremos. Para proceder a realizar las mediciones por medio del Osciloscopio.

Se espera que la medición proporcione un pulso en bajo de valor constante o casi constante del orden del 1 a 2µs. Las flechas rojas indican el pulso y el recuadro muestra la medición del delta de tiempo de cada tarea / delta de tiempo entre la activación y el comienzo de la ejecución.

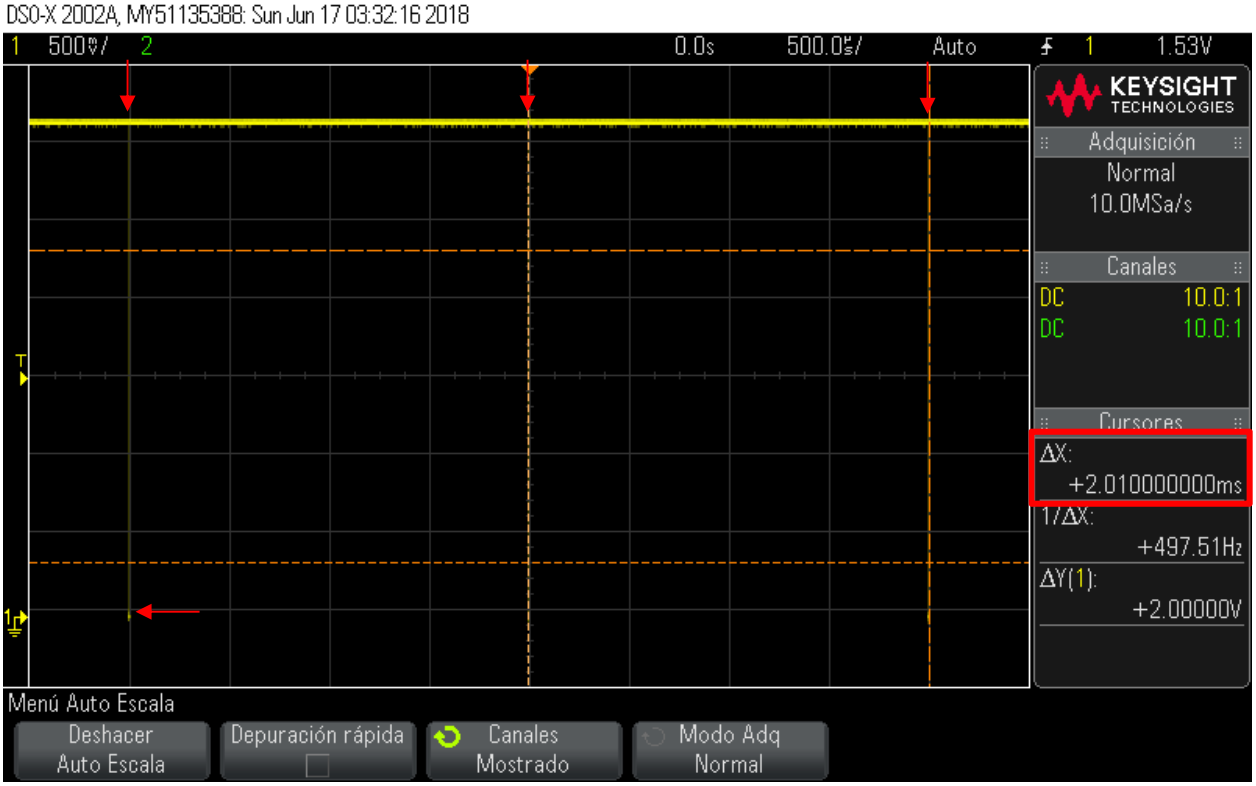
- **Tarea 1ms**
  Delta de tiempo de la tarea:

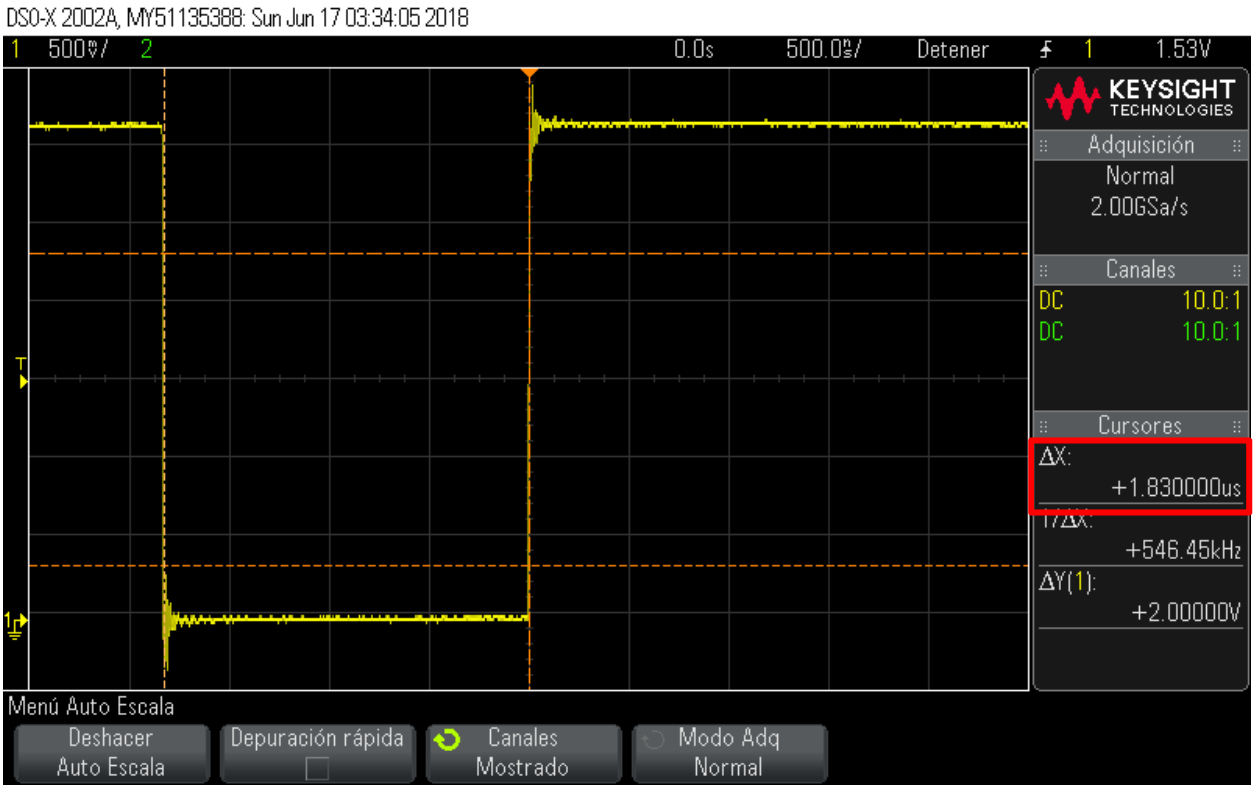


Delta de tiempo entre la activación y ejecución:

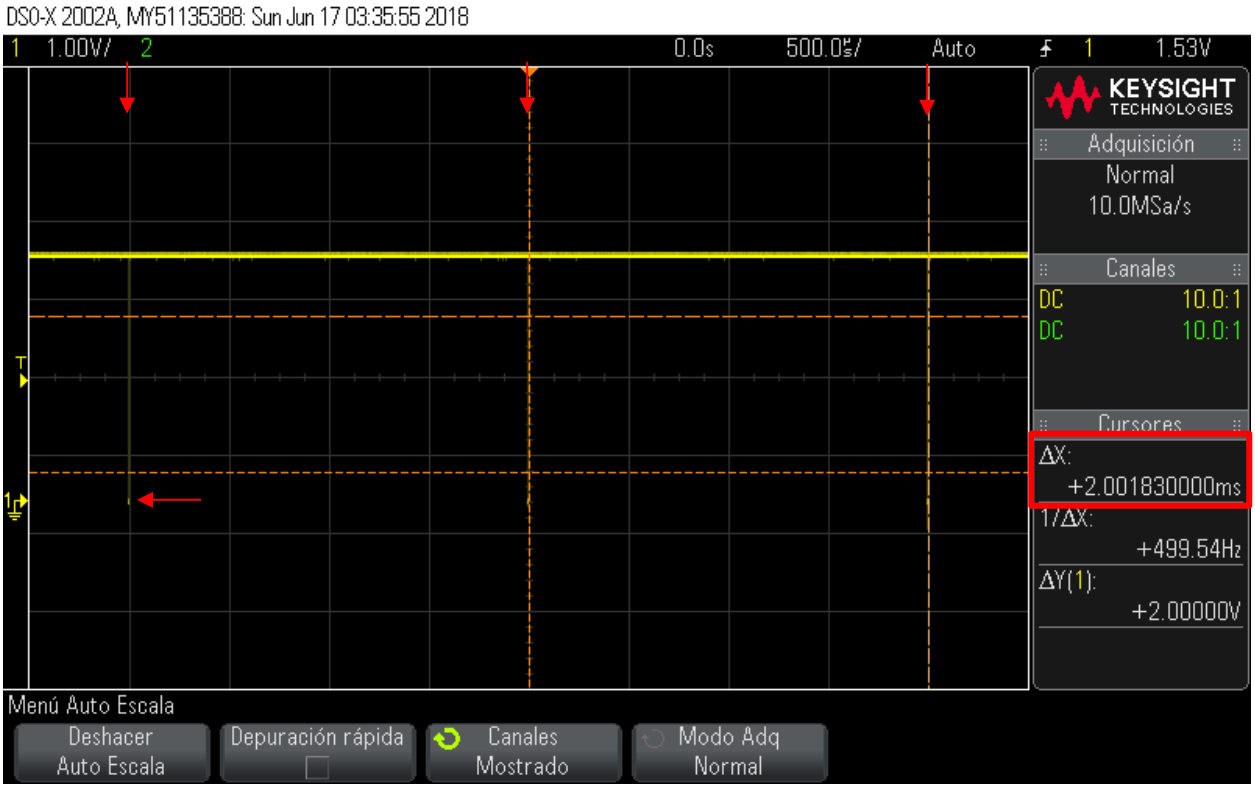


- **Tarea 2ms_a**
  Delta de tiempo de la tarea:

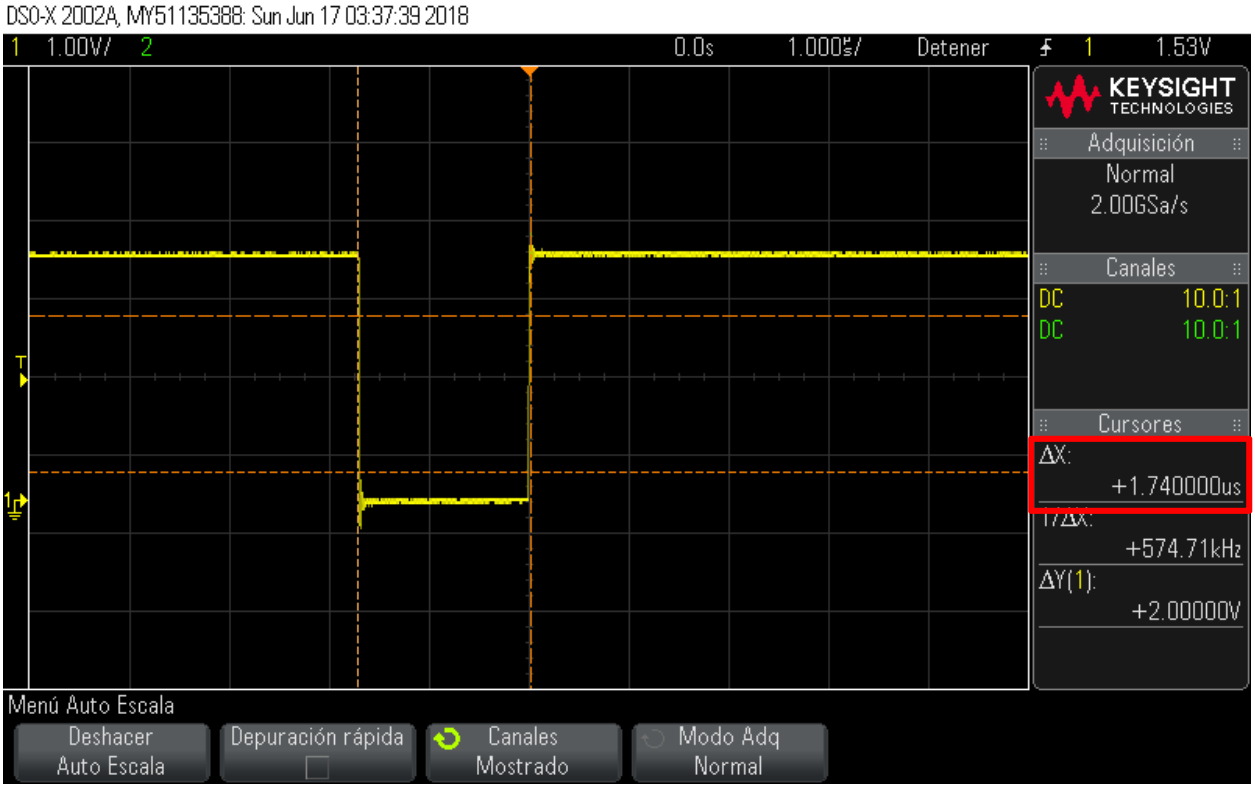




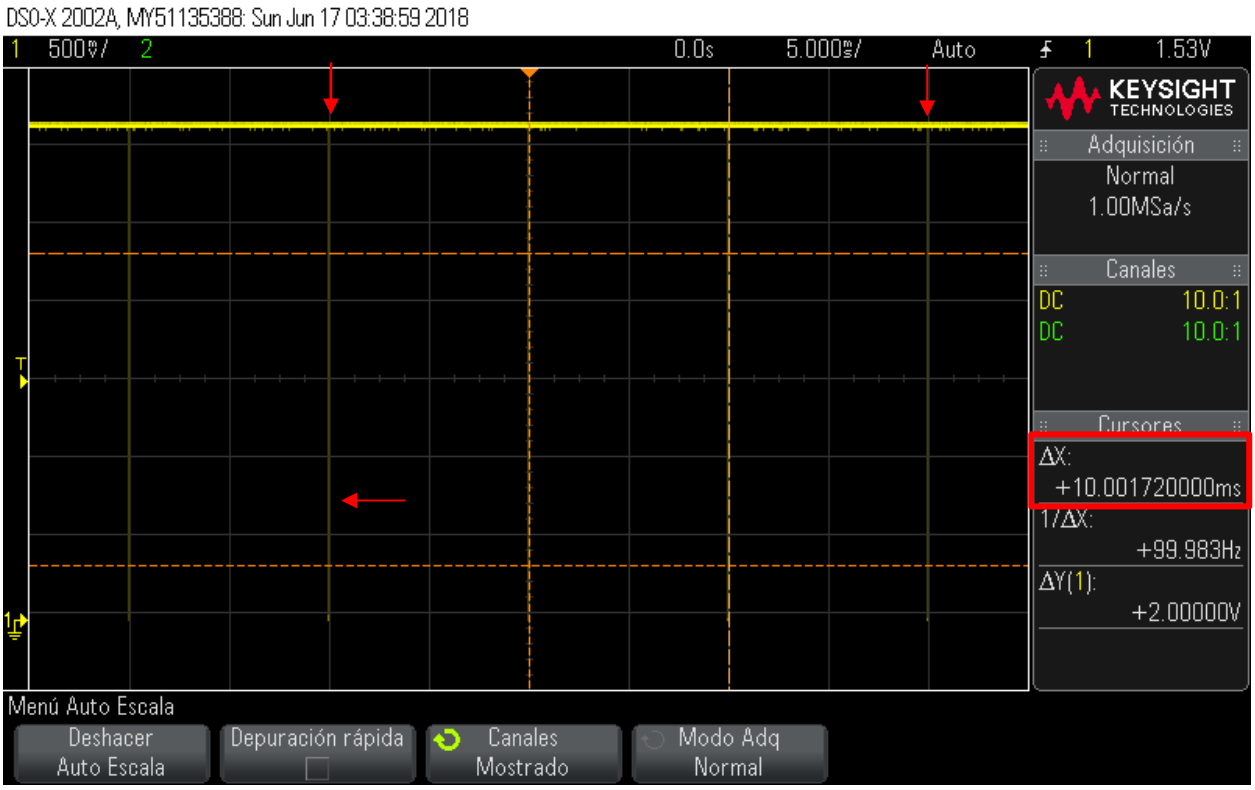
- **Tarea 2ms_b:**
  Delta de tiempo de la tarea:



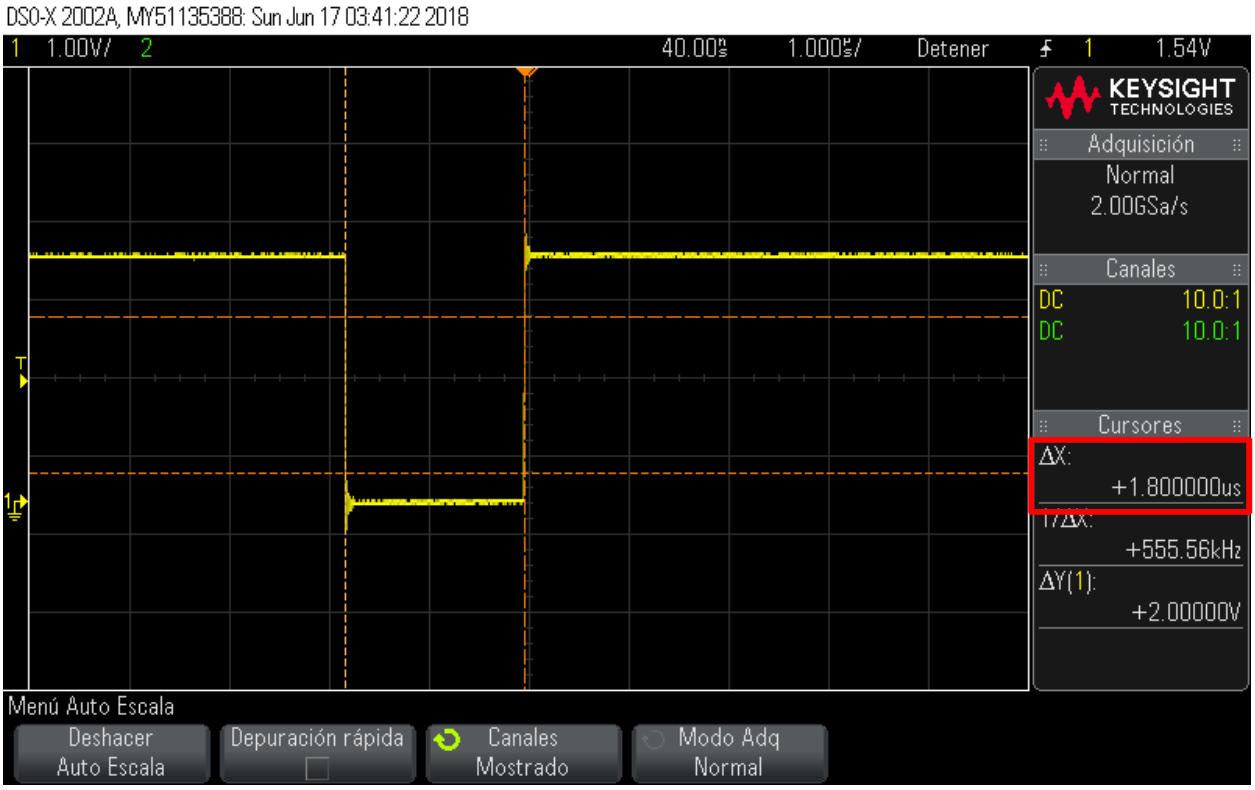
Delta de tiempo entre la activación y ejecución:



- **Tarea 10ms:**
  Delta de tiempo de la tarea:

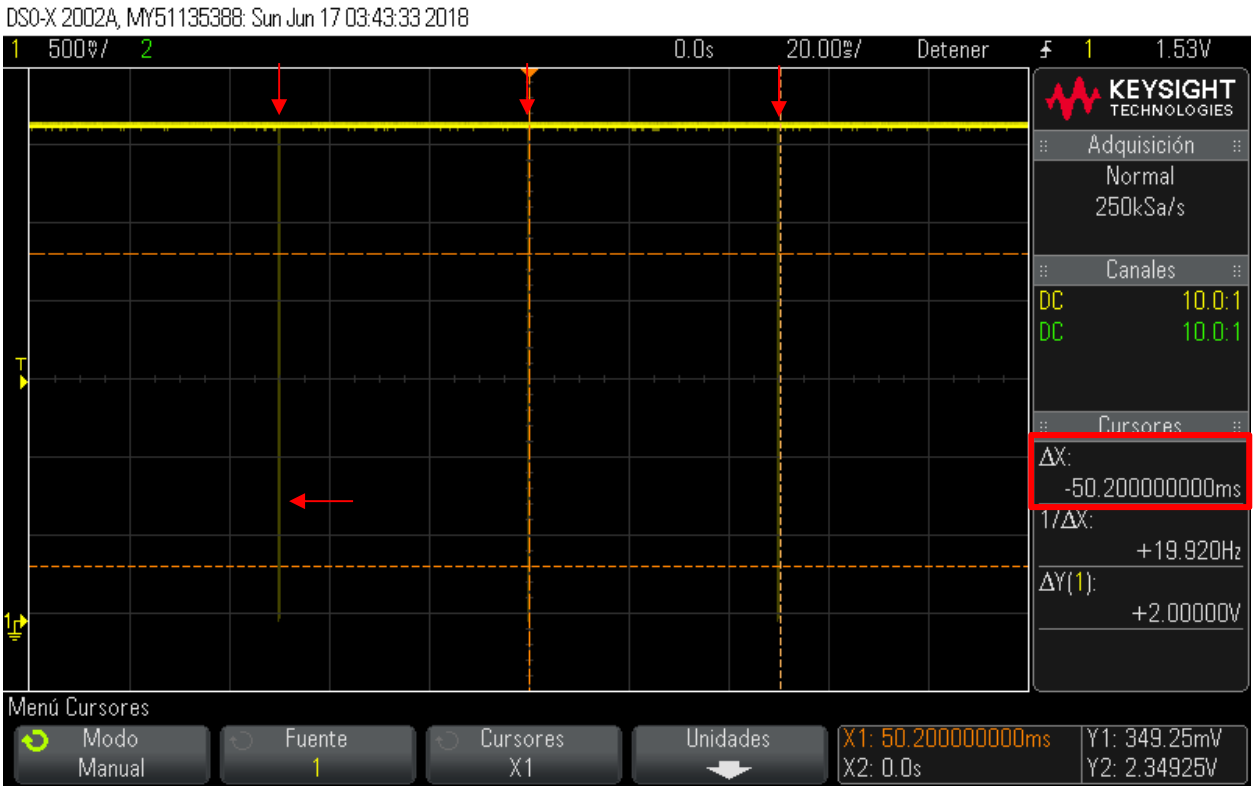


Delta de tiempo entre la activación y ejecución:

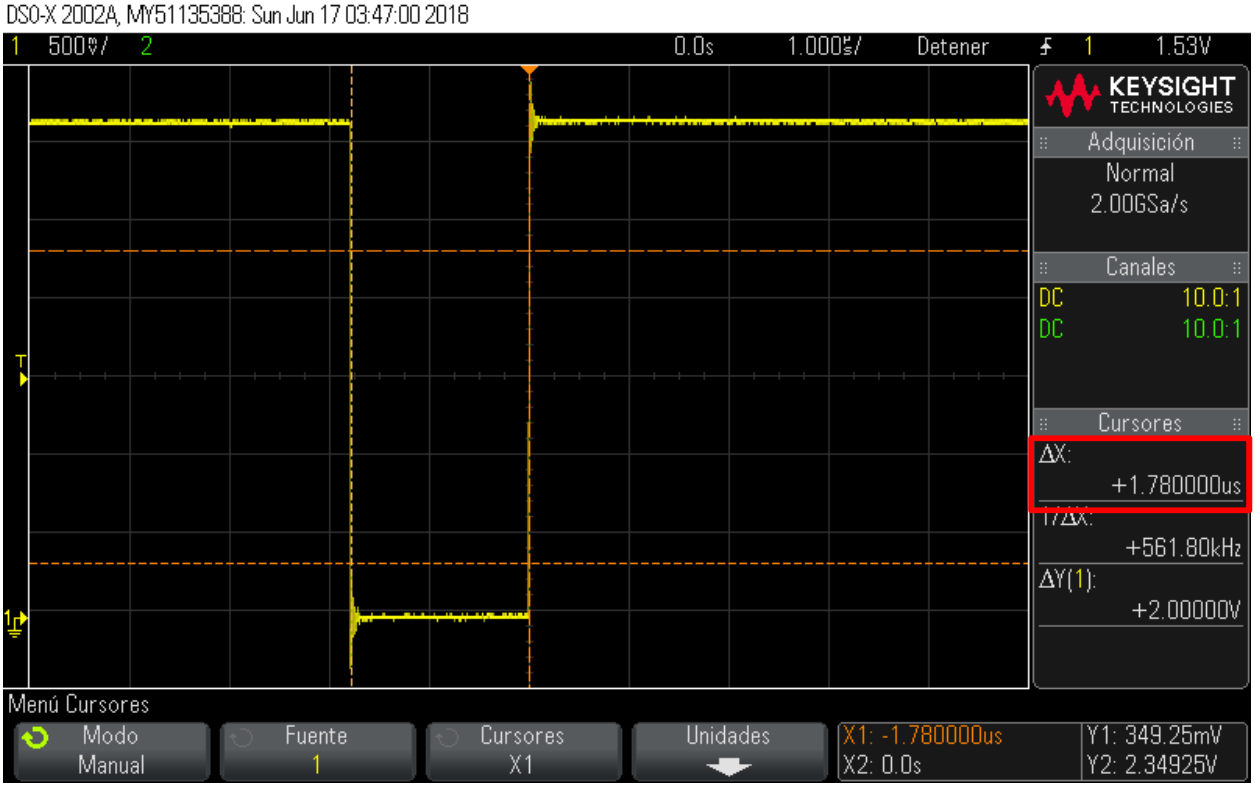


- **Tarea 50ms:**

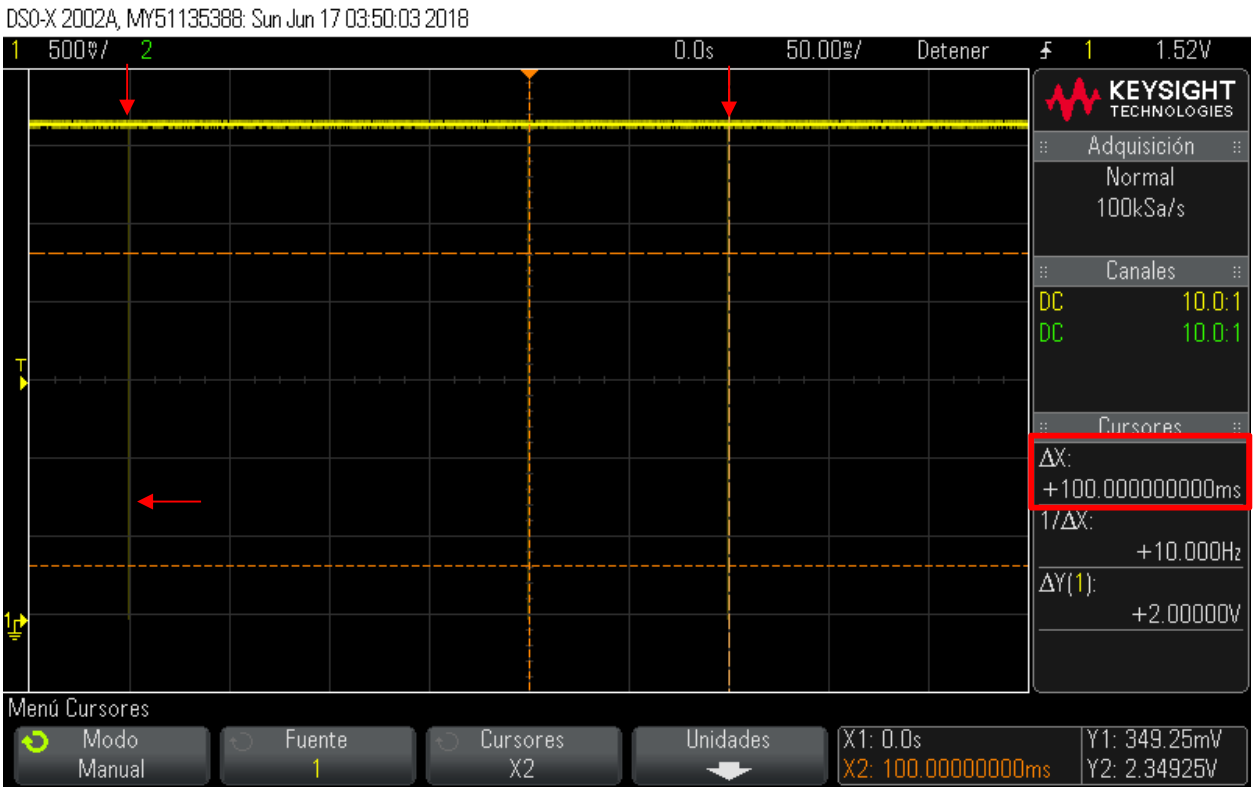
Delta de tiempo de la tarea:



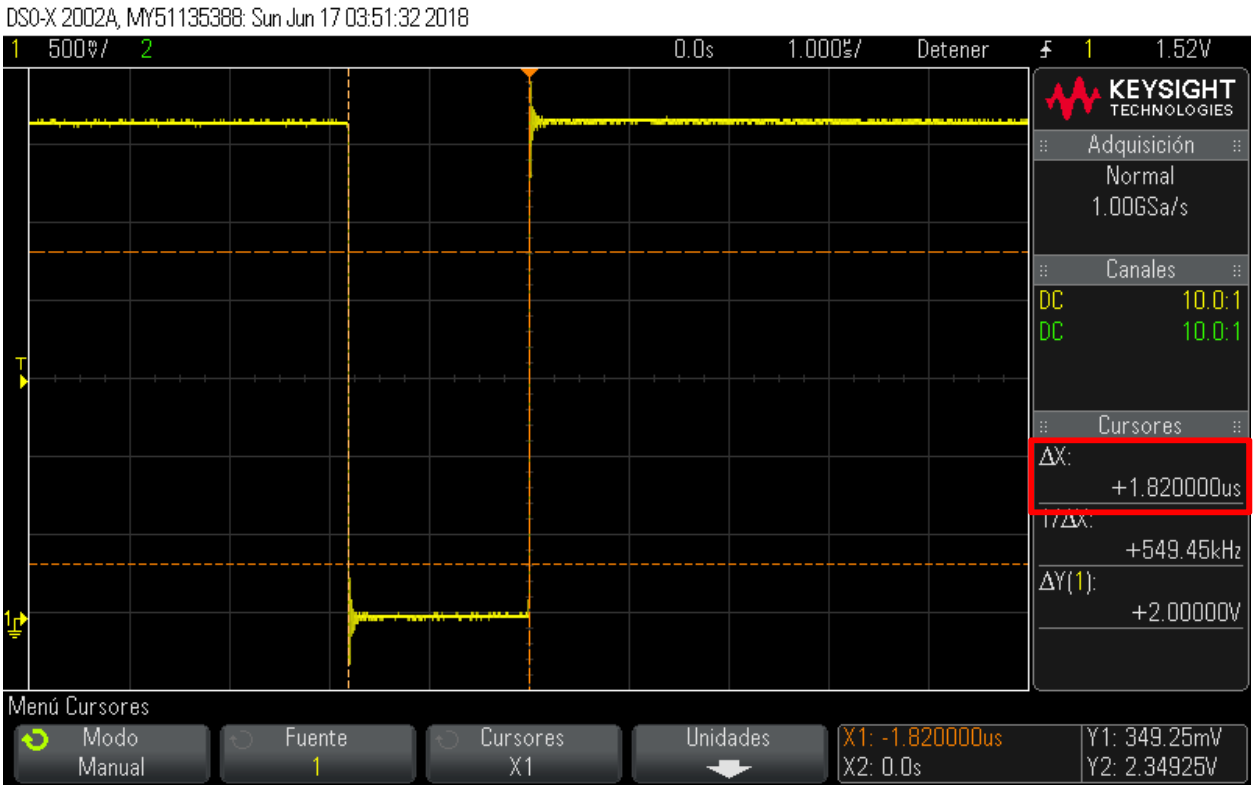
Delta de tiempo entre la activación y ejecución:



- **Tarea 100ms:**
  Delta de tiempo de la tarea:



  Delta de tiempo entre la activación y ejecución:



*Anexo al reporte se adjunta el proyecto con los archivos (mencionados anteriormente) en los cuales se realizaron las modificaciones pertinentes para llevar a cabo la implementación de la tarea.

Conclusión

Mediante el desarrollo de ésta tarea quedó más claro el concepto de Tarea, así como cada una de las partes que conforman el ciclo de vida de cada tarea. Aprendimos a identificar cada uno de los estados (Idle / Ready / Running) y cada uno de los eventos (Activate / Start / Terminate) dentro del código del scheduler que revisamos previamente en clase.

Y aunque al principio todavía existían dudas acerca de los objetivos de la tarea, éstos fueron clarificados en la sesión de asesoría, en la que se nos orientó aún más sobre qué se esperaba de la tarea (el valor esperado del delta de tiempo entre la activación y la ejecución de cada una de las tareas).

También tuve que investigar un poco más sobre el layout de la board, ya que necesitaba información relacionada con el LED en el que se iba a ver reflejada la medición, pin que le pertenecía y la activación del mismo.

Al final, lo que más tiempo me llevó fue realizar las mediciones, ya que había que estar haciendo modificaciones al código para cada una de las tareas.

Referencias

Task Scheduling

https://www.embedded.com/design/operating-systems/4443042/Tasks-and-scheduling

Scheduling Algorithms

http://www.just.edu.jo/~tawalbeh/cpe746/slides/Scheduling_Algorithms.pdf

Compuertas Lógicas Colector Abierto

http://www.ingmecafenix.com/electronica/compuertas-logicas-colector-abierto/

SAM V71 Xplained Ultra_design_documentation_release_rev12.pdf

Atmel-42408-SAMV71-Xplained-Ultra_User-Guide.pdf

Apuntes de clase.